ONLINE COURSE DELIVERY SYSTEM USING SPRING AND STRUTS FRAMEWORK

Sriram Akurati
B.E., Vasavi College of Engineering, Osmania University, India, 2005

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SPRING
2010

ONLINE COURSE DELIVERY SYSTEM USING SPRING AND STRUTS FRAMEWORK

A Project

by

Sriram Akurati

Approved by:

_____, Committee Chair
Dr.Jinsong Ouyang

_____
Date

_____, Second Reader
Dr.Ying Jin

_____
Date

Student: Sriram Akurati

I certify that this student has met the requirements for format contained in the University format manual, and that this Project is suitable for shelving in the Library and credit is to be awarded for the Project.

_____ _____

Dr.Cui Zhang, Graduate Coordinator Date

Department of Computer Science

Abstract

of

ONLINE COURSE DELIVERY SYSTEM USING SPRING AND STRUTS FRAMEWORK

by

Sriram Akurati

The rapid development of computers and telecommunications technologies, along with the introduction of the Internet, provided the tools to enhance distance education and transform the university educational delivery system. The main objective of "Online Course Delivery System" is to deliver courses online, overcoming the geographical and time constraints associated with the traditional programs. It meets the requirements of individuals who want to earn college credits and/or a degree while maintaining a full time job.

In the current project, Spring 2 Framework, Struts 2 Framework along with Java Persistence API are used to develop rich web interfaces, lightweight and reusable components. The customized application developed will enable instructor to post the various course materials, assignments, and grades for students. Also, instructor can maintain security by giving access to only authorized students. Similarly, the student can access the course materials; submit assignments and answer exams posted online.

This project is implemented using Struts 2 which provides parameter binding, data validation and presentation rendering, Spring 2 for data access and transaction management and Java Persistence API for managing object/relational mapping with the database.

_____, Committee Chair
Dr.Jinsong Ouyang

Date: _____

ACKNOWLEDGMENTS

I take this opportunity to thank all the people who are responsible for the successful completion of this project.

I express my sincere gratitude to Dr. Jinsong Ouyang for giving me an opportunity to work under his guidance. This project helped me learn something very new, which I have not learned in my coursework.

I would also like to thank Dr.Ying Jin for being my second reader and helping me with this project.

TABLE OF CONTENTS

Page

Acknowledgments vi

List of Figures xi

Chapter

1. INTRODUCTION 1

2. REQUIREMENT ANALYSIS 3

2.1 Problem Statement 3

2.2 Objective of the Project 3

2.3 Assumptions and Dependencies 3

2.4 Non Functional Requirements 4

2.5 Functional Requirements 4

2.5.1 Use Case: Create Instructor Account 4

2.5.2 Use Case: Delete Instructor's or Student's Account 5

2.5.3 Use Case: Create New Course 6

2.5.4 Use Case: Create New Assignments and Exams 7

2.5.5 Use Case: Delete Assignments or Exams 8

2.5.6 Use Case: Approve Student's Course Registration 8

2.5.7 Use Case: Grade Report View 9

3. ARCHITECTURE AND SYSTEM DESIGN 11

3.1 Architecture Overview 11
3.1.1 Presentation Layer 12
3.1.2 Service Layer 12
3.1.3 Persistence Layer 13
3.1.4 Domain Model Layer 13
3.2 Application Design 14
3.3.1 Administrator 14
3.3.2 Instructor 14
3.3.3 Student 14
3.3 Design Implementation 15
3.3.1 Struts 2 Framework 15
3.3.2 Spring 2 Framework 17
3.3.2.1 Core Container 18
3.3.2.2 Dependency Injection 18
3.3.2.3 Aspect Oriented Programming 19
3.3.3 Object Relational Mapping (ORM) Integration Module 20
3.3.4 Java Persistence API 21
3.4 CkEditor 22
3.5 MySQL 22
4. IMPLEMENTATION 23

4.1 Getting Familiarized with MyEclipse Integrated Development Environment 23

4.2 Integration of Struts 2, Spring 2 and JPA 24

4.2.1 Web.xml Configuration 26

4.2.2 JPA Configuration 27

4.2.3 Entity Class 28

4.2.4 Service Class Interface 34

4.2.5 Service Class Implementation 36

4.2.6 Struts Action Class 43

4.2.7 Spring Configuration File 50

4.2.8 Struts Configuration File 51

4.2.9 JSP Pages 51

4.2.10 Validation Framework 53

4.3 CkEditor Implementation 54

5. PROCEDURAL GUIDE 56

5.1 Instructor 56

5.1.1 Request for Account Creation 56

5.1.2 Creating Courses 58

5.1.3 Uploading Lecture Slides 59

5.1.4 Create New Assignment 59

5.1.5 Edit Assignment 60

5.1.6 Student Approvals 61
5.1.7 View Submitted Assignments 61
5.1.8 Create Exams 62
5.1.9 Edit Exams and View Submitted Exams 63
5.1.10 View Submitted Exams 63
5.1.11 Grade Report 63
5.2 Student 64
5.2.1 Create an Account 65
5.2.2 Register for Course 65
5.2.3 Submit Assignments 66
5.2.4 Take Online Exam 67
5.2.4 Student Grade Report 68
5.3 Administrator 69
5.3.1 Approve Instructor's Account 70
5.3.2 Delete Instructor's and Student's Accounts 70
6. FUTURE WORK AND SUMMARY 71
Bibilography 73

LIST OF FIGURES

Page

1. Figure 2.1 Create Instructor Account 5
2. Figure 2.2 Delete Instructor's or Student's Account 6
3. Figure 2.3 Create New Course 6
4. Figure 2.4 Create New Assignment and Exam 7
5. Figure 2.5 Delete Assignments and Exams 8
6. Figure 2.6 Approve Student's Course Registration 9
7. Figure 2.7 Grade Report View 10
8. Figure 3.1 Architecture 12
9. Figure 3.2 MVC Architecture 16
10. Figure 3.3 Interceptors, OGNL and Valuestack 17
11. Figure 3.4 AOP Layer over Business Objects 20
12. Figure 3.5 DAO Implementation and Access 21
13. Figure 3.6 CkEditor 22
14. Figure 4.1 Snapshot of the IDE 24
15. Figure 4.2 Validation Check 54
16. Figure 5.1 Instructor Account Registration 57
17. Figure 5.2 Instructor Account Activation 57
18. Figure 5.3 Instructor Create Course 58
19. Figure 5.4 Upload Lecture Slides 59

20. Figure 5.5 Create New Assignment 60
21. Figure 5.6 View Student Assignment 62
22. Figure 5.7 Create New Exam 63
23. Figure 5.8 Grade Report 64
24. Figure 5.9 Student Register Course 66
25. Figure 5.10 Student Submit Assignments 67
26. Figure 5.11 Student Submit Exam 68
27. Figure 5.12 Student Grade Report 69

Chapter 1

INTRODUCTION

Earning education is important in both personal and professional life. It is also considered the biggest investment in an individual's life. In earlier days, earning education required attending colleges and schools and required commuting from different geographical locations. But with the onset of internet in every corner of the globe, earning education has taken a different curve.

According to the U.S Department of Education online learning is defined as "the acquisition of knowledge and skills through mediated information and instruction. It encompasses all technologies and supports the pursuit of lifelong learning to all". [5]
It is becoming increasingly popular in schools and colleges where a student or a working adult need not travel the entire distance to learn. He can use the internet in any location and learn online with the greatest benefit of convenience and flexibility. The online class can be open anytime and can be reached with an internet connection.

This project is an effort to develop one such online course delivery system using Spring 2, Struts 2 and JPA. The application developed enables instructor to post the various course materials, assignments, exams and grades for students. At the other end, the student can access the course materials, submit assignments and answer time bound exams posted online.

The report is further organized as follows:

Chapter 2 provides requirement analysis. Chapter 3 deals with the architecture and design part of the application. Chapter 4 illustrates the implementation of the application. Chapter 5 serves as the user-manual for the application. The report is concluded with the future work and summary in Chapter 6.

Chapter 2

REQUIREMENT ANALYSIS

2.1 Problem Statement

Traditionally education has only been provided in classrooms. To earn education students had to commute to schools and colleges. But with the onset of faster internet and broadband, education can not only be earned in school but at a convenient location at home. Working adults who need to earn an additional degree need no longer travel greater distances.

2.2 Objective of the Project

The project aims to build an Online Course Delivery application which enables an instructor to post course materials, exams and assignments online. The students can access these materials, submit assignments and take time bound exams at their own convenience. The project is implemented using Spring Framework, Struts2 Framework and Java Persistence API.

2.3 Assumptions and Dependencies

It is assumed that the students and instructor have at least a broadband internet connection and preferably use Mozilla or Google chrome.

2.4 Non Functional Requirements

1. The project should be developed using Spring Framework, Struts Framework and should include a persistence API.
2. The database included should be MySQL.
3. Rich Text Editor should be used in creating assignment and exams.

2.5 Functional Requirements

1. Create instructor account when requested. Only administrator can approve it.
2. The administrator only should be allowed to delete instructor and student accounts including their contents.
3. Give the instructor the authority to create new courses with appropriate deadlines.
4. The instructor can only create new assignment, exams and upload course materials.
5. Only the instructor should delete assignments, course material and exams.
6. The instructor should approve students who register for his/her courses.
7. The instructor should upload a grade report for each student in his courses.
8. Student should view assignments, exams and course material and submit them online.
9. The student should be able to view grade report posted by the instructor.

2.5.1 Use Case: Create Instructor Account

This use case describes how to the instructor’s account can be created.

Preconditions

1. The instructor is connected to the internet.
2. Valid credentials are entered in the registration page.

Basic Flow

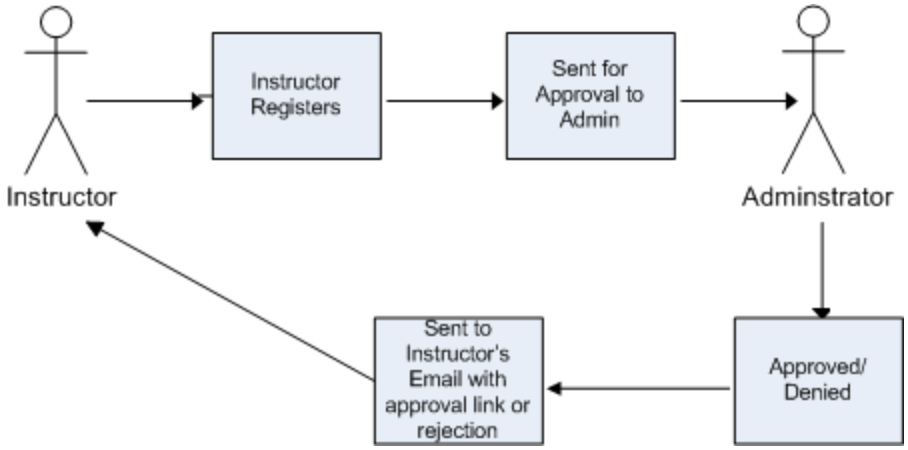

Figure 2.1 Create Instructor Account

Post Conditions

1. Confirmation or declination mail is sent to the instructor.
2. The instructor account is accessible upon approval of the admin.

2.5.2 Use Case: Delete Instructor's or Student's Account

This use case describes how the instructor can delete an instructor or student's account.

Preconditions

1. The administrator should be logged onto the system.
2. The instructor's or student's account should exist.

Basic Flow

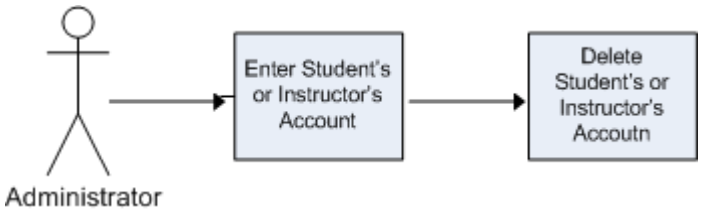

Figure 2.2 Delete Instructor's or Student's Account

Post Conditions

1. The instructor's or student's account is deleted.
2. All the contents of instructor's or student's account are deleted.

2.5.3 Use Case: Create New Course

This use case describes how the instructor creates new courses.

Preconditions

1. To create a new course a valid instructor account should exist.
2. Valid course credentials are entered in the course form fill page.

Basic Flow

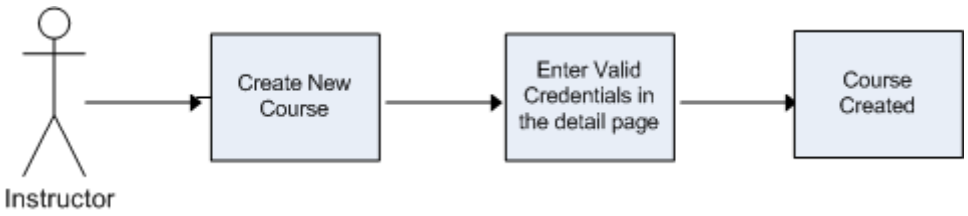

Figure 2.3 Create New Course

Post Conditions:

1. A course is created and is created. New exams and assignments can be uploaded into the system.
2. The course is available for students to register.

2.5.4 Use Case: Create New Assignments and Exams

This use case shows how to create new assignments and exams for the instructor.

Preconditions:

1. The instructor is logged onto the system.
2. The course is created.
3. Valid credentials are filled like due dates and cut off dates.

Basic Flow

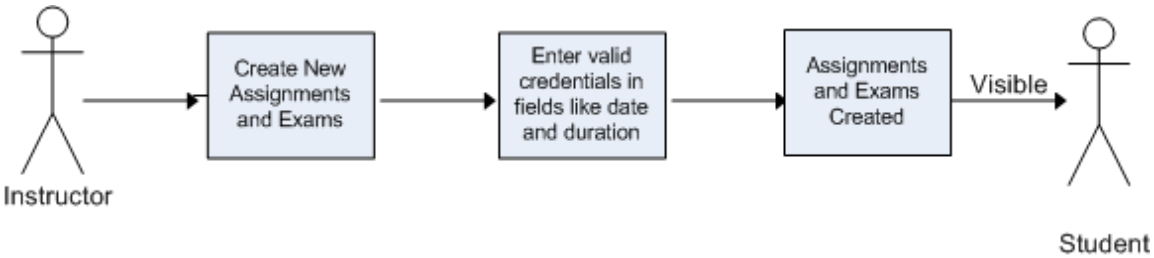

Figure 2.4 Create New Assignment and Exam

Post Conditions:

1. Assignment or exams are created.
2. The student is able to view the assignment or exam.

2.5.5 Use Case: Delete Assignments or Exams

This use case describes how assignments and exams can be deleted.

Preconditions:

1. The instructor is logged onto the system.
2. The assignments and exams should have been created by the user.

Basic Flow

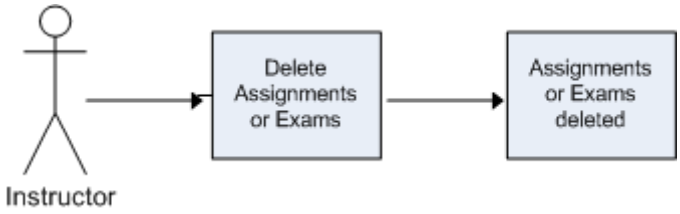

Figure 2.5 Delete Assignments and Exam

Post Conditions:

1. The instructor's assignments or exams as well as the student's assignments deleted.
2. The instructor's assignment's or exam's grade report details are also deleted for the individual students.

2.5.6 Use Case: Approve Student's Course Registration

This use case details on how a student's request for registration is approved by the instructor.

Preconditions:

1. The course is created by the instructor.
2. The student has a valid account.
3. The student should have registered for the course and an approval is pending with the instructor.

Basic Flow

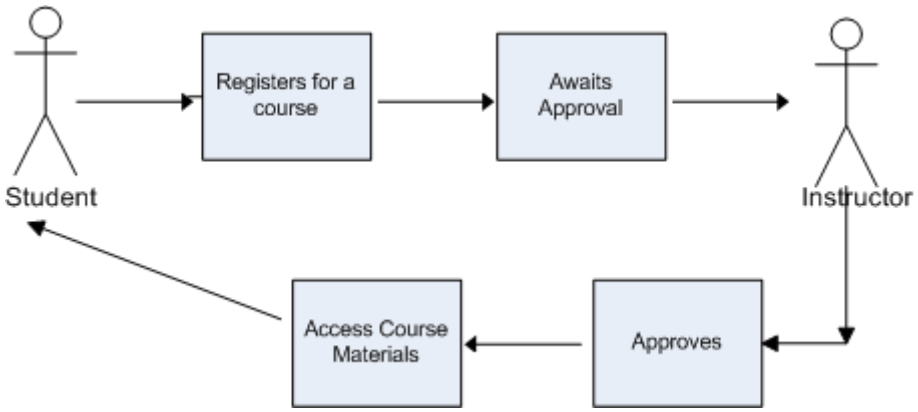

Figure 2.6 Approve Student's Course Registration

Post Conditions:

1. Students can view course materials including assignments, lecture slides and take time bound exams.

2.5.7 Use Case: Grade Report View

This use case highlights on how a student's report can be updated.

Preconditions:

1. A student should have a valid account.

2. Assignment and exams should be created for grade report to be updated.

Basic Flow

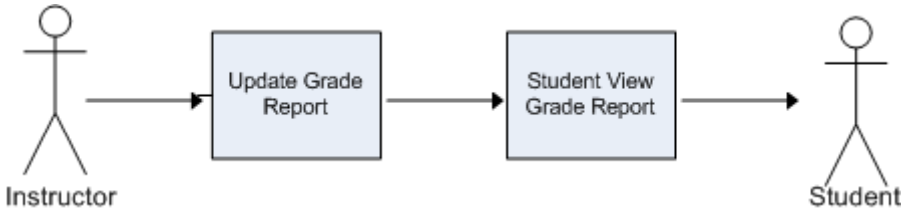

Figure 2.7 Grade Report View

Post Conditions:

1. The student can view the grade report.

Chapter 3

ARCHITECTURE AND SYSTEM DESIGN

The internet contains a lot of dynamic data and the data is constantly updated stored and delivered online using web applications. A web application is a software application that runs over the internet. With the improvements in speed of the internet more complex web applications started emerging and thereby increasing the man hours spend in programming. To reduce programming time and increase reusability or productivity a structure needs to be followed like a framework. "A web application framework is a piece of structural software that provides automation of common tasks of the domain as well as a built-in architectural solution that can be easily inherited by applications implemented on the framework."[1]

3.1 Architecture Overview

Online Course Delivery Project combined well known frameworks like Spring, Struts and JPA to achieve loose coupling. The project made use of four different layers presentation layer, service layer, persistence layer and the domain model layer. Each layer has its own unique purpose and are isolated from other layers.

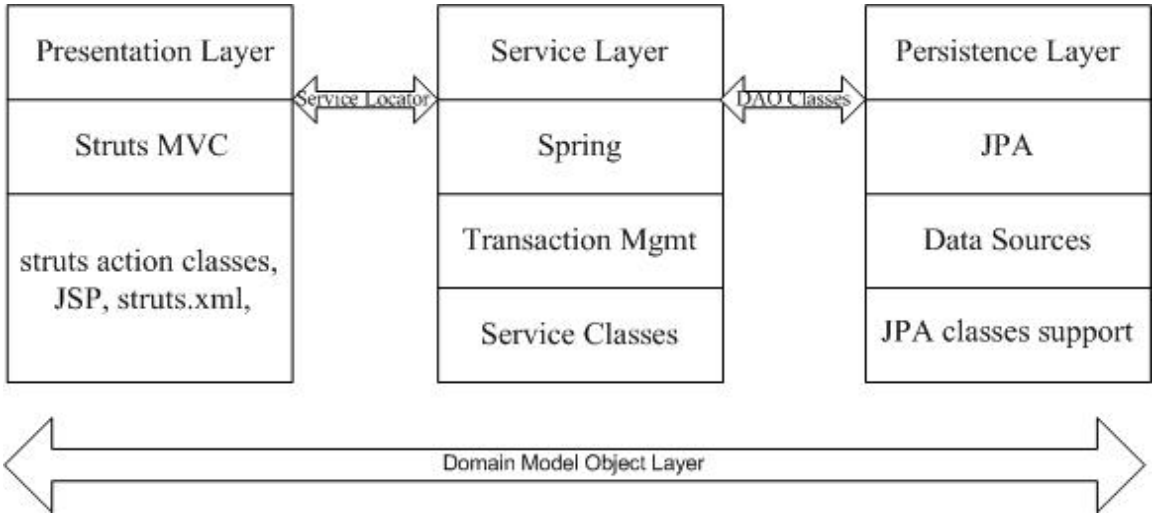

Figure 3.1 Architecture

3.1.1 Presentation Layer

Any typical web application consists of a presentation layer. In this project Struts 2 provides the presentation layer. The presentation layer in this project is responsible for:

- Managing requests and responses for the user.
- Assembling a model that can be presented in a view.
- Performing client side validation.
- Call appropriate service layer classes.

3.1.2 Service Layer

The middle component of a web application is generally the service layer. The service layer is typically found around in the presentation layer or in the persistence layer. This does not achieve the concept of loose coupling and can be overcome using dependency

injection (also known as inversion of control). The service layer in this project is Spring 2.0. In spring framework a XML file wires all the objects that contain references to transaction handling and service objects that include DAO (data access objects). The service layer in this project is responsible for the following:

- Managing transactions.
- Communicating with the persistence layer.

3.1.3 Persistence Layer

The persistence layer's job is to provide an interface for storage mechanisms. The API should be abstract and independent of the storage technology or vendor. The following list contains the type of code in persistence layer:

- Store and retrieve objects
- Save, Update, Create and delete information on tables.
- Query Support.

3.1.4 Domain Model Layer

Communication between service, persistence and domain model takes place in the form of objects. In online course delivery the database information is read as objects which can be presented to the presentation layer.

3.2 Application Design

The application was designed in keeping three types of users in purview, the administrator, instructor and the student. Each user has different privileges and actions.

3.3.1 Administrator

The administrator can perform the following tasks:

- Approve an instructor account whenever requested.
- Delete an instructor account and all its contents.
- Delete a student's account and its contents.

3.3.2 Instructor

The instructor can do the following tasks:

- Send a request access to the administrator for creating a new account.
- Create course with details and registration deadlines.
- Grant course access to students when requested.
- Create and delete time bound exams, assignments with deadlines and lecture notes online.
- Grade exams, assignments and maintain a grade sheet for each sheet.

3.3.3 Student

The student can perform the following tasks:

- Request permission for registering to a course.
- View and submit assignments based on deadlines.
- Take time bound exams posted by the instructor.
- View Lecture Slides posted by the instructor.
- View grade report put by the instructor.

3.3 Design Implementation

Struts 2 Framework is an established MVC Framework and Spring Framework is a lightweight container which makes wiring of objects easier. Integration of Spring and Struts brings in a new design approach that solves existing problems with java applications. Spring is an open source framework with new features like dependency injection and aspect oriented programming which are aimed in making the application loosely coupled, lightweight and easily testable. Spring makes use of plain-vanilla JavaBeans to achieve things that were possible with EJBs [1].

3.3.1 Struts 2 Framework

Struts 2 is an apache's open source framework for developing java enterprise edition applications which implements the Model View Controller (MVC) design pattern. Struts 2 has more advantages over Struts1 with features like interceptors for layering cross cutting concerns, annotation configuration over xml, Open-Graph Navigational Language (OGNL) and a better tag API. MVC architecture can be used to separate cross cutting concerns. The controller in it accepts all the incoming requests from the browser and

sends them to the specific action or the model. The model contains the business logic which processes the request from the controller and renders the result to the view as shown in Figure 3.2 which is Figure 1.3 from [1].

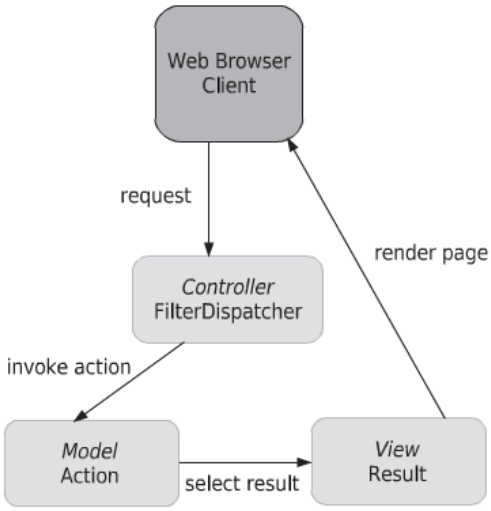

Figure 3.2 MVC Architecture

The other important features of Struts 2 which make MVC architecture cleaner are the valuestack, interceptor and the OGNL. Once the controller calls the appropriate action the action is passed through a set of interceptors to the result. The interceptor are assigned to do work like validation, session management etc. Depending on the programmer's logic the interceptors can called be before and after the action is performed and are supposed to do a set of tasks. On the other hand the data that is transferred during the processing of a request is stored in what the Struts 2 calls the valuestack. "Data is moved to the valuestack in preparation for request processing, it is manipulated there during action

execution, and it is read from there when the results render their response pages" [1] as shown in Figure 3.3 which is Figure 1.4 from [1]. And the OGNL is a language for manipulating the valuestack data.

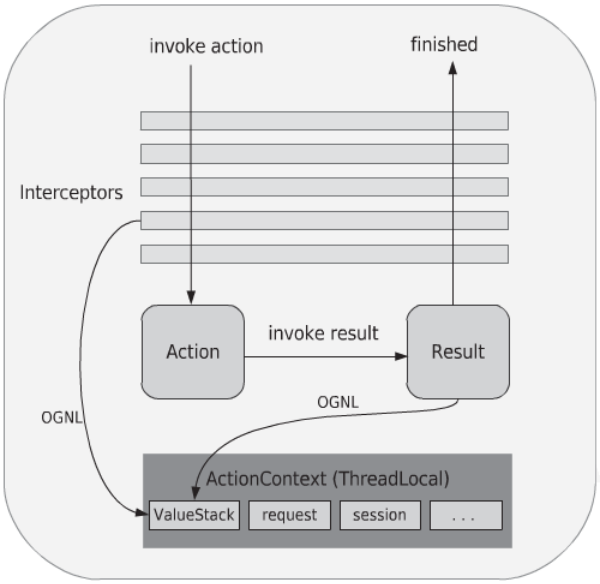

Figure 3.3 Interceptors, OGNL and Valuestack

3.3.2 Spring 2 Framework

Spring Framework is an open source framework application that addresses the complexity of the enterprise application development. Spring makes its applications use (Plain Old Java Objects) POJOs with a primary goal of reducing complexity. The most interesting aspect of Spring is the Dependency Injection and Aspect Oriented Programming. Spring is a layered architecture consisting of different modules. Some of the modules in Spring are:

3.3.2.1 Core Container

Spring's core container provides the basic functionality of the framework. All the modules work on the basis of the core container. The core container purpose is to contain and manage the lifecycle of the application objects [6].

3.3.2.2 Dependency Injection

The Spring container implements a loose coupling technique called dependency injection. Dependency Injection is a concept wherein a method requiring an object need not look for an object but in turn the method is provided the object. In Spring the objects instead of looking for dependencies are provided objects during instantiation by the container. Applicationcontext is one implementation of the container in Spring which loads beans depending on the configurations given in the file. Loading an application context from an xml file is shown

```
ApplicationContext context = new FileSystemXmlApplicationContext("xyz.xml");
```

The beans in the xml files will not be initiated until needed. To retrieve a bean getBean() method is passed with the name of the bean as shown below:

```
MyBean myBean = (MyBean) factory.getBean("mybean");
```

Spring provides two major types of dependency injection setter injection and constructor injection. The xml file xyz.xml shown below contains a bean id and the class name and in this case both the setter property and constructor property.

```
<bean id=”mybean” class=”com.ocd.Instructor”>
<property name=”course” ref=”csc201”/>
<constructor-arg value=”csc201”/>
```

3.3.2.3 Aspect Oriented Programming

Aspect Oriented Programming is a technique that allows programmers to modularize cross cutting concerns [6]. Cross cutting concerns are supporting functions such as logging, security. These functions often find their way into business logic which increases the complexity of the code and duplicates the code. In the figure show below the business objects are also involved with system services. So each object’s responsibility is not only to perform the business logic but they have to take care of other services like logging and security.

To overcome these problems AOP makes it easy to separate the cross cutting concerns and modularize them. AOP is a blanket that contains all the system services which covers other business objects as shown in Figure 3.4 which is Figure 1.8 from [6].

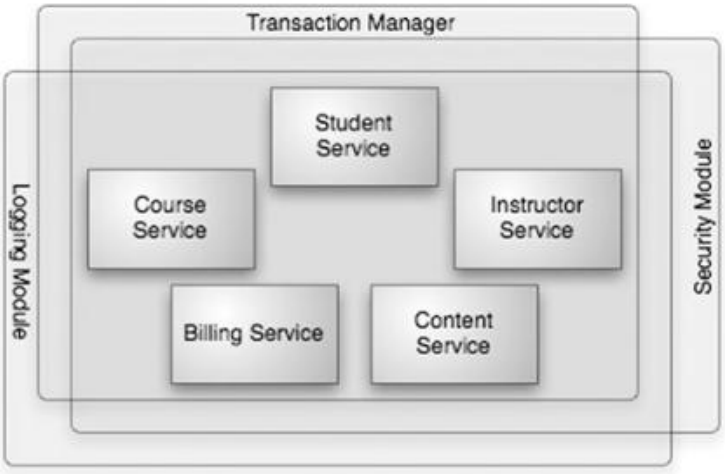

Figure 3.4 AOP Layer over Business Objects

3.3.3 Object Relational Mapping (ORM) Integration Module

Rather than using jdbc for data access Spring also supports integration to various ORM products such as JPA, Hibernate and iBATIS. But integrating with Spring provides an advantage compared to integrating it directly in areas like session management, resource management and transaction management. Spring's ORM integration with other persistent tools is built on DAO support which makes it easier to switch to different persistent frameworks. Data Access Objects (DAO) provides means to read and write data to the database. They expose their functionality only through interfaces. And service objects can access DAO implementation through the DAO interface as shown in figure 3.5.

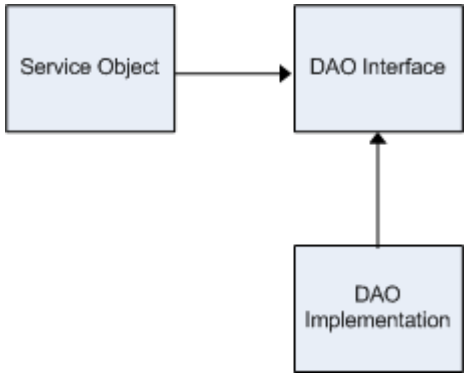

Figure 3.5 DAO Implementation and Access

Spring Framework provides abstraction for transaction management. Programmatic via callback template and declarative transaction which was available to EJB but now has been provided to POJOs. [2]

3.3.4 Java Persistence API

Java Persistence API is a specification from Sun which was released under JEE 5. The Java Persistence API provides an object/relational mapping facility for managing relational data in java applications. Persistence is a way of automatically mapping normal java objects into a database. Rather than writing cumbersome insert, update, select and delete commands, the framework will take care of this through automation. Persistent data in JPA are referred to as entities. An entity in a database can be stored and retrieved as a whole.

3.4 CkEditor

CkEditor is an open source WYSIWYG inline rich editor used in web pages. Similar to Microsoft Word editing of text or html is possible in web pages. Ckeditor contains common editing functions like choosing font, font size, making characters bold, hyper linking etc… With minimal programming knowledge ckeditor can be inserted into any website and provide them with advanced tools for editing content. Ckeditor is written javascript and is compatible with most browsers. A picture of Ckeditor is shown in figure 3.7.

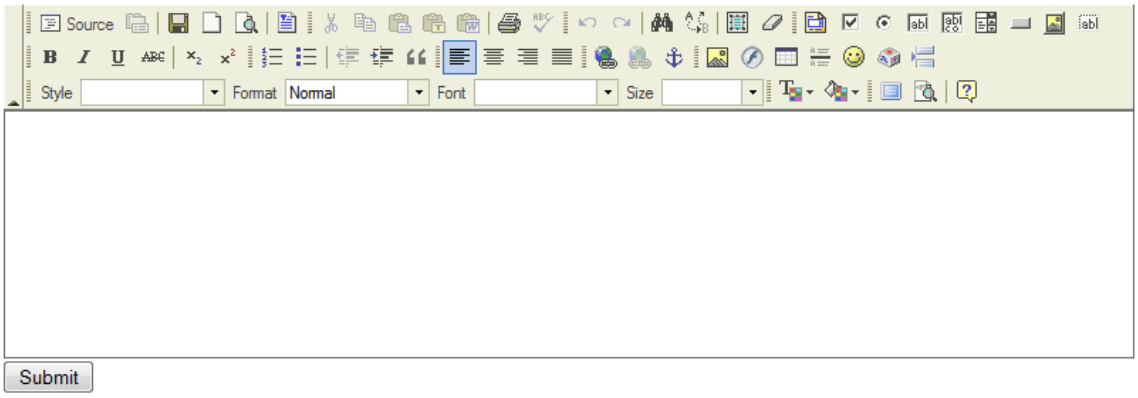

Figure 3.6 CkEditor

3.5 MySQL

MySQL is a free to download database without any commercial licensing involved [3]. MySQL 4.1 is being used in this project.

Chapter 4

IMPLEMENTATION

This chapter provides an overview of how the Struts 2, Spring 2 and JPA are integrated into one project. The chapter also provides an overview of CkEditor implementation in the project.

4.1 Getting Familiarized with MyEclipse Integrated Development Environment

MyEclipse IDE can be downloaded http://www.myeclipseide.com/. MyEclipse contains an integrated Tomcat 6 application server and the latest version of JDK, which need not be installed separately. A web project from file -> new can be selected to create a project. The IDE is shown in Fig 4.1 The IDE consists of three parts:

- Package Explorer
- Workbench
- Platform Runtime

Package Explorer

The package explorer shows a tree view of the project.

Workbench

The workbench is the development environment where resources can be created and managed.

Platform Runtime

The platform runtime gives the output of created resources.

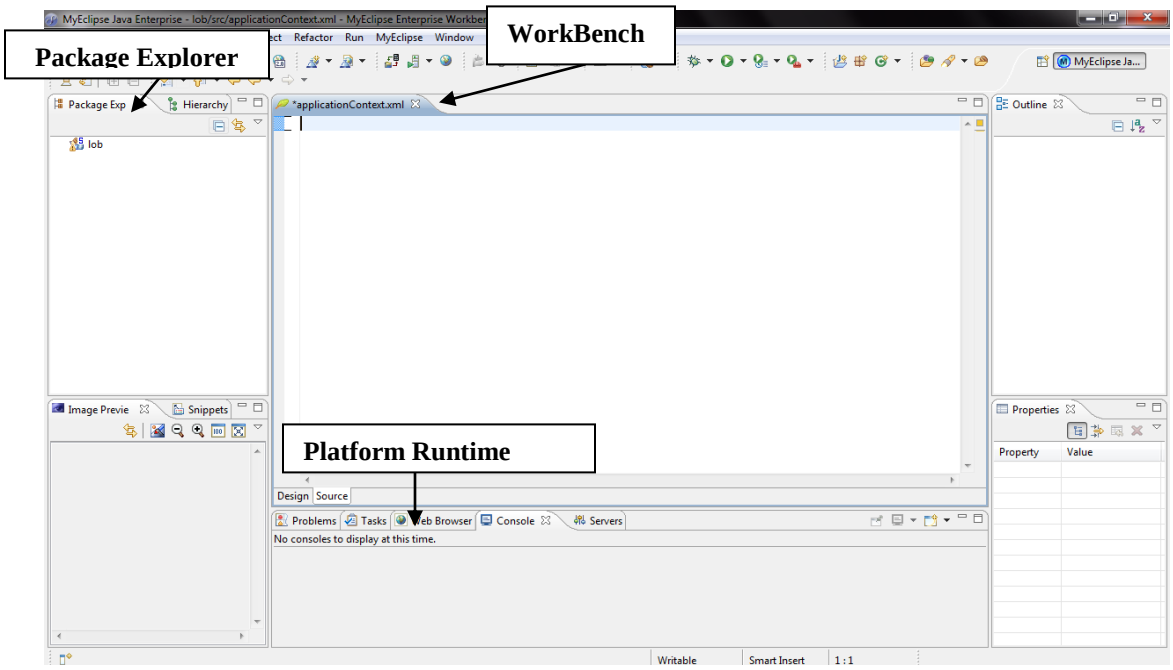

Figure 4.1 Snapshot of the IDE

4.2 Integration of Struts 2, Spring 2 and JPA

Once the web project is created it will contain two main folders src and WebContent. The jar files or the dependencies need to be added to the project. The dependencies should be stored in \WebContent\WEB-INF\lib. This can be done by right clicking on the lib folder selecting import and choosing a folder where the dependencies are stored. Dependencies can be downloaded from:

http://struts.apache.org/2.x/

http://www.springframework.org/download

http://dev.mysql.com/get/Downloads/Connector-J

The following are the dependencies used for this project:

- Commons-fileupload-1.2.1.jar
- Commons-io.1.3.2.jar
- Commons-logging-1.0.4.jar
- Freemarker-2.3.8.jar
- Java-core-2.5.jar
- Jta.jar
- Mysql-connector-java-5.1.6-bin.jar
- Persistence.jar
- Slf4j-api-1.5.8.jar
- Sl4fj-simple-1.5.8.jar
- Spring-jpa.jar
- Struts2-core-2.0.14.jar
- Struts2-spring-plugin-2.0.14.jar
- Toplink-essentials.jar
- Xwork-2.0.7.jar

In the later version of MyEclipse Struts 2 , Spring 2 and JPA can be easily added by right clicking on the project, clicking MyEclipse and adding all of the above capabilities.

4.2.1 Web.xml Configuration

Once the dependencies are added. The web.xml in \WebContent\WEB-INF\web.xml should be configured. The web.xml file should contain the following:

```
 <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
       org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
</filter>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<session-config>
    <session-timeout>240</session-timeout>
</session-config>
<listener>
    <listener-class>
```

```
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
```

The web.xml file will make the container redirect all the requests to FilterDispatcher with home page as index.jsp and spring's ContextLoaderListener is configured as context listener. The web.xml also contains a session timeout for the web application. In this project the web application session time out is 240 minutes.

4.2.2 JPA Configuration

To add JPA to the project two things need to be done. One is adding dependencies. Dependencies can vary from one vendor to another vendor. In this project mysql is being used so Mysql-connector-java-5.1.6-bin.jar is added. The other is adding a persistence.xml file. In \src\META-INF create a persistence.xml file and set its content to the following:

```
<persistence-unit name="OCDPU" transaction-type="RESOURCE_LOCAL">
        <provider>oracle.toplink.essentials.PersistenceProvider</provider>
         <class>com.ocd.jpa.login.Ocdlogin</class>
         <properties>
        < property name = "toplink.jdbc.driver" value="com.mysql.jdbc.Driver"/>
        <property name = "toplink.jdbc.url" value="jdbc:mysql://locahost:3306/DB"/>
        <property name = "toplink.jdbc.user" value = "user"/>
        <property name = "toplink.jdbc.password" value = "password"/>
```

<property name = “toplink.cache.shared.Ocdstudent” value=”false”/>

</properties>

</persistence-unit>

In this project oracle toplink JPA is used so each and every entity class that is used in the project should also be specified in the persistence.xml like the Ocdlogin java file with its entire path. The driver values and the url should be entered depending on the db vendor. In the case of MySQL the name of the database used should be entered in place of DB.

4.2.3 Entity Class

The entity class is defined in the src path with the @Entity annotation and with getter and setter items to every field of the table that is being mapped. In the later versions of MyEclipse entity code can be generated by right clicking on the project, clicking on MyEclipse and pressing Generate Entities & DAOs. The code snippet for ocdexam is shown below:

```
package com.ocd.jpa.inst;
import java.math.BigDecimal;
import java.util.Date;
import javax.persistence.*;

@Entity
@Table(name = "OCDEXAM", schema = "sampdb")
```

```
public class Ocdexam implements java.io.Serializable {

	// Fields

	private BigDecimal id;
	private String coursenumber;
	private BigDecimal examnumber;
	private String examcontent;
	private Date examdatetime;
	private byte[] attachment;
	private String filename;
	private String maxmarks;
	private BigDecimal duration;

	// Constructors

	/** default constructor */
	public Ocdexam() {
	}
	/** minimal constructor */
	public Ocdexam(BigDecimal id) {
		this.id = id;
```

```
}

/** full constructor */
public Ocdexam(BigDecimal id, String coursenumber, BigDecimal examnumber,
        String examcontent, Date examdatetime, byte[] attachment,
        String filename, String maxmarks, BigDecimal duration) {
    this.id = id;
    this.coursenumber = coursenumber;
    this.examnumber = examnumber;
    this.examcontent = examcontent;
    this.examdatetime = examdatetime;
    this.attachment = attachment;
    this.filename = filename;
    this.maxmarks = maxmarks;
    this.duration = duration;
}

// Property accessors
@Id
@Column(name = "ID", unique = true, nullable = false, precision = 22, scale = 0)
public BigDecimal getId() {
    return this.id;
```

```
}

public void setId(BigDecimal id) {
        this.id = id;
}

@Column(name = "COURSENUMBER", length = 255)
public String getCoursenumber() {
        return this.coursenumber;
}

public void setCoursenumber(String coursenumber) {
        this.coursenumber = coursenumber;
}

@Column(name = "EXAMNUMBER", precision = 22, scale = 0)
public BigDecimal getExamnumber() {
        return this.examnumber;
}

public void setExamnumber(BigDecimal examnumber) {
        this.examnumber = examnumber;
```

```
}

@Column(name = "EXAMCONTENT", length = 255)
public String getExamcontent() {
	return this.examcontent;
}

public void setExamcontent(String examcontent) {
	this.examcontent = examcontent;
}

@Temporal(TemporalType.TIMESTAMP)
@Column(name = "EXAMDATETIME", length = 7)
public Date getExamdatetime() {
	return this.examdatetime;
}

public void setExamdatetime(Date examdatetime) {
	this.examdatetime = examdatetime;
}

@Column(name = "ATTACHMENT")
```

```
public byte[] getAttachment() {
        return this.attachment;
}

public void setAttachment(byte[] attachment) {
        this.attachment = attachment;
}

@Column(name = "FILENAME", length = 255)
public String getFilename() {
        return this.filename;
}

public void setFilename(String filename) {
        this.filename = filename;
}

@Column(name = "MAXMARKS", length = 255)
public String getMaxmarks() {
        return this.maxmarks;
}
```

```
	public void setMaxmarks(String maxmarks) {
		this.maxmarks = maxmarks;
	}

	@Column(name = "DURATION", precision = 22, scale = 0)
	public BigDecimal getDuration() {
		return this.duration;
	}

	public void setDuration(BigDecimal duration) {
		this.duration = duration;
	}

}
```

The variable examcontent is a string which stores all the data from the rich text editors. The attachment for each exam is converted into bytes and then stored in the database in a longblob field. The entity also contains constructors which can be accessed from other interfaces. Examdatetime is a date variable which can store the date and time of the exam.

4.2.4 Service Class Interface

The service class interface contains all the methods for CRUD Create, Read, Update and Delete actions. This is a common technique where implementation details are hidden

which keeps the beans loosely coupled. The code snippet for ocdexam interface is shown below:

```
package com.ocd.jpa.inst;
import java.math.BigDecimal;
import java.util.Date;
import java.util.List;
public interface IOcdexamDAO {

        public void save(Ocdexam entity);
        public void delete(Ocdexam entity);
        public Ocdexam update(Ocdexam entity);
        public Ocdexam findById(BigDecimal id);
        public List<Ocdexam> findByProperty(String propertyName, Object value);
        public List<Ocdexam> findByCoursenumber(Object coursenumber);
        public List<Ocdexam> findByExamcontent(Object examcontent);
        public List<Ocdexam> findByAttachment(Object attachment);
        public List<Ocdexam> findByFilename(Object filename);
        public List<Ocdexam> findByMaxmarks(Object maxmarks);

        public List<Ocdexam> findByTwoProperties(String propertyName, final Object
        value,  String propertyName1, final Object value1);
```

```
        public List<Ocdexam> findAll();
}
```

4.2.5 Service Class Implementation

The service class implementation implements the CRUD operations using Java Persistence API. All the transactions in the project are declarative transactions with the use of a simple annotation @Transactional. The code snippet for save and delete are shown below:

```
package com.ocd.jpa.inst;
import java.math.BigDecimal;
import java.util.Date;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceException;
import javax.persistence.Query;
import org.springframework.context.ApplicationContext;
import org.springframework.orm.jpa.JpaCallback;
import org.springframework.orm.jpa.support.JpaDaoSupport;
import org.springframework.transaction.annotation.Transactional;
@Transactional
public class OcdexamDAO extends JpaDaoSupport implements IOcdexamDAO {
```

```
// property constants
public static final String COURSENUMBER = "coursenumber";
public static final String EXAMCONTENT = "examcontent";
public static final String ATTACHMENT = "attachment";
public static final String FILENAME = "filename";
public static final String MAXMARKS = "maxmarks";
public void save(Ocdexam entity) {
        logger.info("saving Ocdexam instance");
        try {
                getJpaTemplate().persist(entity);
                logger.info("save successful");
        } catch (RuntimeException re) {
                logger.error("save failed", re);
                throw re;
        }
}

public void delete(Ocdexam entity) {
        logger.info("deleting Ocdexam instance");
        try {
                entity = getJpaTemplate().getReference(Ocdexam.class,
                                entity.getId());
```

```
                getJpaTemplate().remove(entity);
                logger.info("delete successful");
        } catch (RuntimeException re) {
                logger.error("delete failed", re);
                throw re;
        }
}

public Ocdexam update(Ocdexam entity) {
        logger.info("updating Ocdexam instance");
        try {
                Ocdexam result = getJpaTemplate().merge(entity);
                logger.info("update successful");
                return result;
        } catch (RuntimeException re) {
                logger.error("update failed", re);
                throw re;
        }
}

public Ocdexam findById(BigDecimal id) {
        logger.info("finding Ocdexam instance with id: " + id);
```

```
        try {
                Ocdexam instance = getJpaTemplate().find(Ocdexam.class, id);
                return instance;
        } catch (RuntimeException re) {
                logger.error("find failed", re);
                throw re;
        }
}
@SuppressWarnings("unchecked")
public List<Ocdexam> findByProperty(String propertyName, final Object value)
{
        logger.info("finding Ocdexam instance with property: " + propertyName
                        + ", value: " + value);
        try {
                final String queryString = "select model from Ocdexam model
                where model." + propertyName + "= :propertyValue order by
                model.examnumber";
                return getJpaTemplate().executeFind(new JpaCallback() {
                        public Object doInJpa(EntityManager em)
                                        throws PersistenceException {
                                Query query = em.createQuery(queryString);
                                query.setParameter("propertyValue", value);
```

```
                return query.getResultList();
            }
        });
    } catch (RuntimeException re) {
        logger.error("find by property name failed", re);
        throw re;
    }
}

@SuppressWarnings("unchecked")
public List<Ocdexam> findByTwoProperties(String propertyName, final Object
value,  String propertyName1, final Object value1) {
    logger.info("finding Ocdexam instance with property: " + propertyName
            + ", value: " + value);
    try {
        final String queryString = "select model from Ocdexam model
        where model."
                + propertyName + "= :propertyValue and model."
+propertyName1+"=:propertyValue1 order by model.examnumber";
        return getJpaTemplate().executeFind(new JpaCallback() {
            public Object doInJpa(EntityManager em)
                    throws PersistenceException {
```

```
                        Query query = em.createQuery(queryString);
                        query.setParameter("propertyValue", value);
                        query.setParameter("propertyValue1", value1);
                        return query.getResultList();
                    }
                });
        } catch (RuntimeException re) {
                logger.error("find by property name failed", re);
                throw re;
        }
}

public List<Ocdexam> findByCoursenumber(Object coursenumber) {
        return findByProperty(COURSENUMBER, coursenumber);
}

public List<Ocdexam> findByExamcontent(Object examcontent) {
        return findByProperty(EXAMCONTENT, examcontent);
}

public List<Ocdexam> findByAttachment(Object attachment) {
        return findByProperty(ATTACHMENT, attachment);
```

```
}

public List<Ocdexam> findByFilename(Object filename) {
        return findByProperty(FILENAME, filename);
}

public List<Ocdexam> findByMaxmarks(Object maxmarks) {
        return findByProperty(MAXMARKS, maxmarks);
}


@SuppressWarnings("unchecked")
public List<Ocdexam> findAll() {
        logger.info("finding all Ocdexam instances");
        try {
                final String queryString = "select model from Ocdexam model";
                return getJpaTemplate().executeFind(new JpaCallback() {
                        public Object doInJpa(EntityManager em)
                                        throws PersistenceException {
                                Query query = em.createQuery(queryString);
                                return query.getResultList();
                        }
```

```
                });
        } catch (RuntimeException re) {
                logger.error("find all failed", re);
                throw re;
        }
    }

    public static IOcdexamDAO getFromApplicationContext(ApplicationContext ctx)
    {
        return (IOcdexamDAO) ctx.getBean("OcdexamDAO");
    }
}
```

In spring framework and JPA data can be accessed using the JPA template. The persist(entity) method will save the entity in the database. The merge(entity) will update the database with the entity values. The remove(entity) will remove the entity from the database.

4.2.6 Struts Action Class

The struts 2 action class contains an execute() method which is executed only after the properties defined in the action class are populated. The properties can be textfields or textareas from the jsp files that are submitted. In this project the action class also contains

DAO interface beans that are injected into the action class which are defined in applicationContext.xml. The code snippet for the action class:

```
package com.ocd.struts.inst;
import java.math.BigDecimal;
import java.util.Calendar;

import com.ocd.jpa.inst.*;
import com.opensymphony.xwork2.ActionSupport;
public class ExamCreate extends ActionSupport {

public String Save()
{
        byte[] b =null;
        if(getAttachment()!=null){
        b = new byte[(int) getAttachment().length()];
        try {
        FileInputStream fileInputStream = new
        FileInputStream(getAttachment());
        fileInputStream.read(b);
        for (int i = 0; i < b.length; i++) {
        System.out.print((char)b[i]);
```

```
}
} catch (FileNotFoundException e) {
System.out.println("File Not Found.");
e.printStackTrace();
}
catch (IOException e1)
{
System.out.println("Error Reading The File.");
e1.printStackTrace();
}
}
List<Ocdexam> oe =
getOcdexamdao().findByCoursenumber(getCoursenumber());
if(oe.size()!=0)
{
        Iterator it = oe.iterator();
        while(it.hasNext())
        {
                Ocdexam oe1 = (Ocdexam)it.next();
                if(oe1.getExamnumber().equals(getExamNumber()))
                {
```

```
                        System.out.println("Exam Date
                        Time"+oe1.getExamdatetime());
                        if(getAttachment()!=null)
                        {
                                oe1.setAttachment(b);
                                oe1.setFilename(getAttachmentFileName());
                        }
                        oe1.setExamcontent(getEditorDefault());
                        getOcdexamdao().update(oe1);
                }
        }
}
addActionMessage("Your Exam is Saved");
return "success";

}

        //Submit Method
        private File attachment;
        private String attachmentFileName;
        private String attachmentContentType;
        private String FileName;
        private java.util.Date datepicker;
```

```
private BigDecimal Duration;
private String editorDefault;
private String Coursenumber;
private String ExamNumber;

public String getEditorDefault() {
        return editorDefault;
}
public void setEditorDefault(String editorDefault) {
        this.editorDefault = editorDefault;
}
public String getCoursenumber() {
        return Coursenumber;
}
public void setCoursenumber(String coursenumber) {
        Coursenumber = coursenumber;
}
private IOcdexamDAO ocdexamdao;

public IOcdexamDAO getOcdexamdao() {
        return ocdexamdao;
}
```

```
public void setOcdexamdao(IOcdexamDAO ocdexamdao) {
        this.ocdexamdao = ocdexamdao;
}
public BigDecimal getDuration() {
        return Duration;
}
public void setDuration(BigDecimal duration) {
        Duration = duration;
}
public java.util.Date getDatepicker() {
        return datepicker;
}
public void setDatepicker(java.util.Date datepicker) {
        this.datepicker = datepicker;
}
public void setAttachment(File attachment) {
this.attachment = attachment;
}
public String getAttachmentFileName() {
return attachmentFileName;
}
public void setAttachmentFileName(String attachmentFileName) {
```

```
        this.attachmentFileName = attachmentFileName;
        }
        public String getAttachmentContentType() {
                return attachmentContentType;
        }
        public void setAttachmentContentType(String attachmentContentType) {
        this.attachmentContentType = attachmentContentType;
        }
        public String getFileName() {
        return FileName;
        }
        public void setFileName(String fileName) {
        FileName = fileName;
        }
}
```

The class's save() method is executed after the instructor presses the submit button to create an exam. This class contains different variables with their getter and setter functions which retrieve the input from the jsp page. For example the editorDefault variable is a string which holds the content in the rich text editor. The getEditorDefault() will retrieve the string from the editor. The value is stored into the database by calling the update method. Similarly the attachment is also stored by first converting the attachment into byte array and then updating it.

4.2.7 Spring Configuration File

Spring configures its beans in a spring configuration file applicationContext.xml which is located in \WebRoot\WEB-INF. In Spring the configuration is also known as wiring of beans. The code snippet is as below:

```
<bean id="entityManagerFactory"
        class="org.springframework.orm.jpa.LocalEntityManagerFactoryBean">
        <property name="persistenceUnitName" value="OCDPU" />
</bean>
<bean id = "transactionManager" class =
"org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>
        <tx:annotation-driven transaction-manager="transactionManager" />
<bean id="ocdlogindao" class="com.ocd.jpa.inst.OcdExamDAO">
        <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>
```

The configuration file references to the persistence.xml file where the entityManageFactory wires with other classes such as the login class. This project uses declarative transaction management using annotations. To define this <tx:annotation-driven transaction-manager=”transactionManager”/> is used.

4.2.8 Struts Configuration File

The struts configuration file should be placed in \src path following nomenclature as struts.xml. The configuration file contains details as to which action class should fire when a particular url is hit and which results might be chosen by that action to complete processing. The code snippet for struts.xml is shown below:

```
<action name = "InstructorExamCreateFCKEditor" class="com.ocd.struts.inst.Exams"
method="Save">
        <interceptor-ref name="Insstack"></interceptor-ref>
        <result name="success" type="redirect-action">
                <param name="actionName">InstructorExamTab.action</param>
                <param name="Coursenumber">${Coursenumber}</param>
        </result>
</action>
```

In the above case when the action class's Exams save() method returns a success which triggers another action InstructorExamTab action.

4.2.9 JSP Pages

The web content in this project is displayed using jsp pages. Struts2 user interface and data tags are used in them for dynamic content. A code snippet is shown below:

```
<s:set name="username" value="#session.user" />
```

```
<s:if test="#session.loginInstructor != 'true'">
   <jsp:forward page="/index.jsp" />
</s:if>
<div id="global" style="font-family:verdana" align="right"> <s:property
value="username" />   </div>
  <center>
<h3 style="font-family:verdana">Exam Create  </h3>
<h5 style="font-family:verdana"> Exam Number: <s:property value = "ExamNumber"
/></h5>
<s:form action="InstructorExamCreateFCKEditor"  enctype="multipart/form-data"
method="post">
 <div id="global" style="font-family:verdana"><s:actionmessage /></div>
                    <s:hidden name="ExamNumber"/>
                    <s:hidden name="Coursenumber"/>
                    <s:hidden name="ExamDate"/>
                    <s:hidden name="ExamTime"/>
                    <s:hidden name="Duration"/>
                    <FCK:editor instanceName="editorDefault" height="250px"
                    width="1000px"/>
                    <br/>
                              <s:file  name="attachment" label="Attachment" />
                              <s:submit  />
```

```
<br/>
</s:form>
</center>
```

The above html code is a form which generates a rich text editor. The instance of the editor is editorDefault. When submit button is pressed all this property fields as well as the other fields like exam number, course number … are transferred to its respective action class which is the InstructorExamCreateFCKEditor.

4.2.10 Validation Framework

Struts2 provides an easy and a unique method of validating forms. One method of implementation is xml based configuration method. The jsp file which needs to be validated requires its file name to be <actionclass>-validation.xml. login-validation.xml is defined in the project. When the username is not mentioned a message displaying "name is required" as shown in fig 4.1. The code snippet for login-validation.xml is:

```
<validators>
  <field name="name">
    <field-validator type="required">
      <message>name is required.</message>
    </field-validator>
  </field>
  <field name="password">
    <field-validator type="required">
```

```
        <message>password is required.</message>
      </field-validator>
    </field>
</validators>
```

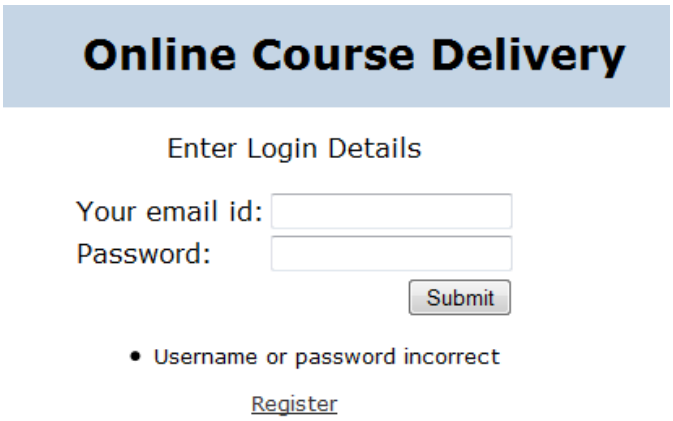

Figure 4.2 Validation Check

4.3 CkEditor Implementation

Ckeditor can be downloaded from http://ckeditor.com/download. Once unzipped the folder fckeditor should be placed in \WebRoot path. The following files should be included in the \WebRoot\WEB-INF\lib:

- Commons-io-1.3.2.jar
- Commons-fileupload-1.2.1.jar

- Imageinfo-1.9.jar
- Java-core-2.5.jar
- Slf4j-api-1.5.8.jar
- Slf4j-simple-1.5.8.jar

Once the files are included insert the following code which will display the inline content editor:

```
<%@ taglib uri="http://java.fckeditor.net" prefix="FCK" %>
<FCK:editor instanceName="editorDefault" height="250px" toolbarSet="Basic">
</FCK:editor>
```

Chapter 5

PROCEDURAL GUIDE

This chapter provides a user manual for the online course delivery project using Spring , Struts2 framework and JPA. The manual is divided into categories of the type of users.

5.1 Instructor

The instructor can perform the following tasks using the current web application:

- Request for Instructor account
- Create courses for student's access.
- Post Lecture notes, Assignments.
- Post time-bound exams
- Save marks obtained by students in the assignments and exams
- Post final grades of students

5.1.1 Request for Account Creation

An instructor has to request Admin to create an account to use the application i.e., to post the courses, exams, assignments etc

To create an account:

1. Click the ***Register*** link on the home page.
2. Fill up the form. The Figure 5.1 is the snapshot of Instructor account.
3. Press the Submit button.

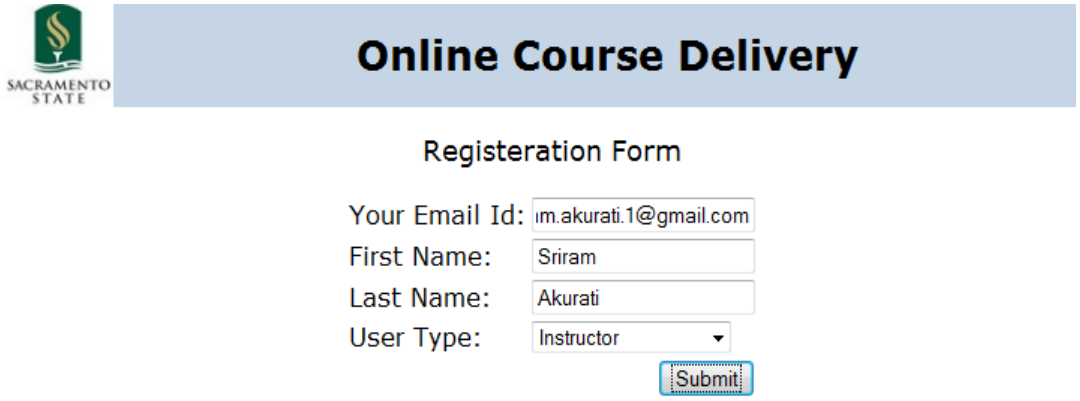

Figure 5.1 Instructor Account Registration

4. After the account instance is created, it is submitted for approval to the administrator. Once the administrator approves it a mail is sent to the user for creating his password. Figure 5.2 shows the password creation and account activation form.

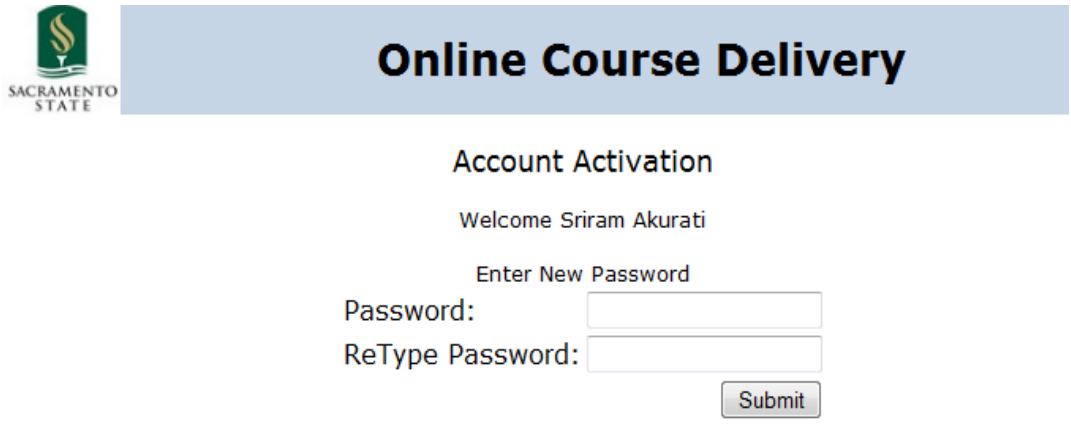

Figure 5.2 Instructor Account Activation

5.1.2 Creating Courses

Instructor can create courses and deliver them to students with assignments, lecture notes and exam in them. The procedure for creating a course is:

1. Click on ***Courses*** tab on the left of the page and click on ***New Course*** link.
2. Fill the form that appears including the registration deadline. Figure 5.3 shows the form to be filled by the instructor for course creation.

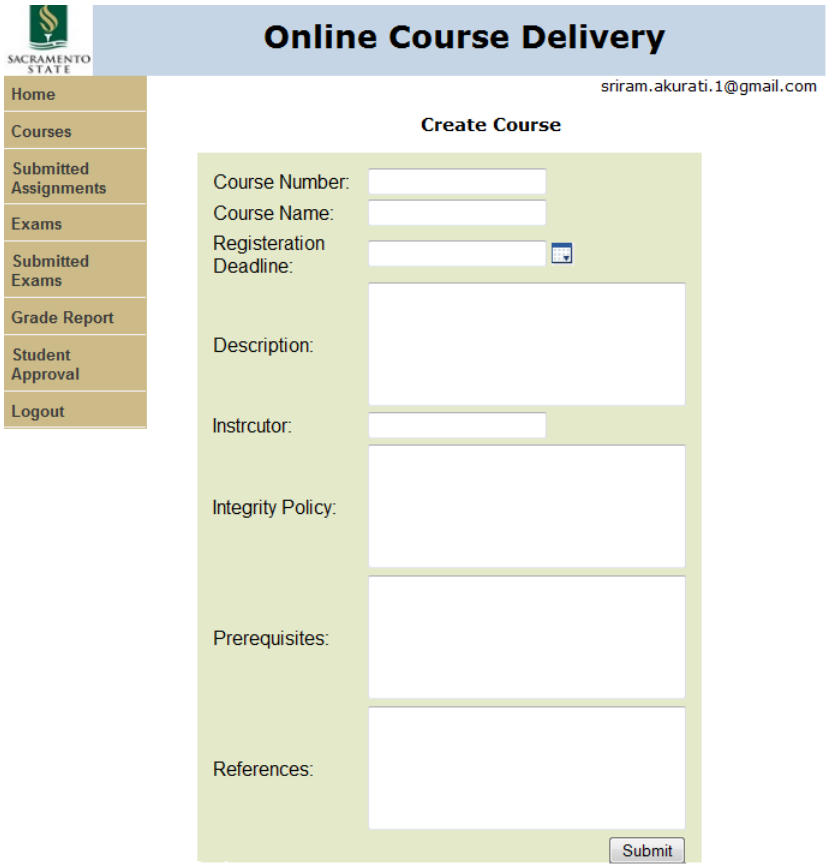

Figure 5.3 Instructor Create Course

5.1.3 Uploading Lecture Slides

Once the course is created the instructor can upload any documents onto his course. The following is the procedure for uploading lecture slides or any other document.

1. Click on ***Courses*** tab on the left of the page and click on the course where the slides need to uploaded.
2. Three tabs with the details, lecture slides and assignments appear. Click on ***Lecture Slides*** and ***Add Attachment*** as shown in Figure 5.4

Figure 5.4 Upload Lecture Slides

5.1.4 Create New Assignment

The instructor can create an assignment with a due date and cutoff date. The student can submit an assignment after the due date if he has not already submitted it. The procedure for creating a new assignment is:

1. Click on ***Courses*** tab on the left of the page and click on the course where the assignment needs to be created.

2. Three tabs with the details, lecture slides and assignments appear. Click on ***Assignments*** link and ***New Assignment*** link.
3. Fill the form which contains details like cutoff date, due date , penalty , marks and grade and click submit.
4. A new window with a rich text editor opens with a browser button where attachments can be added as shown in Figure 5.5.

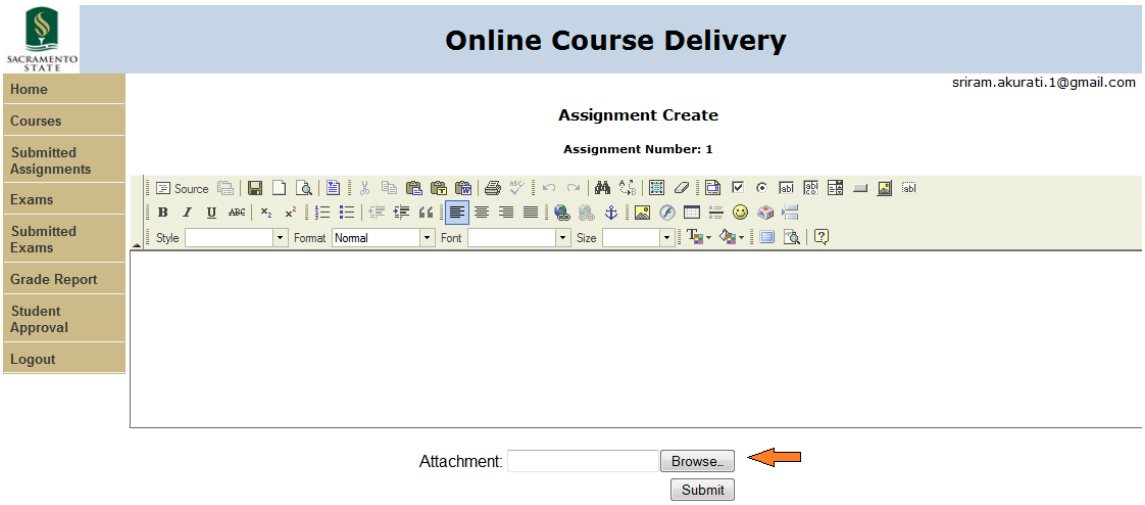

Figure 5.5 Create New Assignment

5.1.5 Edit Assignment

Just as an assignment can be created it can also be edited. All the fields including date fields can be edited. Attachments can be removed and new content can be posted online.

5.1.6 Student Approvals

An instructor's approval is needed for a student to view his course. Once a student registers for a course an automatic approval is sent to the instructor. The steps for this are:

1. Click on ***Student Approval*** link and the ***Course*** link.
2. A list of usernames appears with their respective statuses.
3. Click on the username whose approval is needed.
4. A new page appears where approvals or denial can be made.

5.1.7 View Submitted Assignments

Assignments submitted by students can viewed by the instructor and graded at the same time. The procedure for this is:

1. Click on the ***Submitted Assignments*** Tab on the left and click on the ***Course*** link.
2. A list of assignments created by the instructor appears in a tabular form.
3. Click on the ***Assignments*** link.
4. All the users who have submitted the assignments will appear with the submission date and username.
5. Click on the ***Usernames*** link to view the assignment.
6. The Assignment can be graded immediately and it is reflected on the student's grade report as well as shown in Figure 5.6.

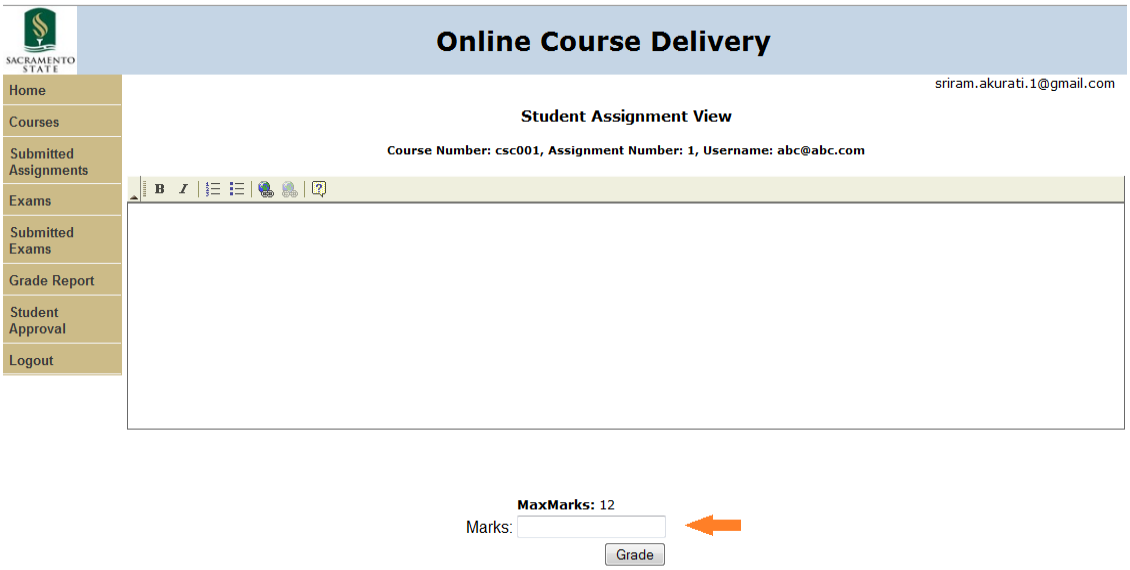

Figure 5.6 View Student Assignment

5.1.8 Create Exams

Time bound exams can be created by the instructor. The instructor can select a time and duration of the exam. Attachments can also be added to the exam if necessary. The student will only be able to open the exam ten minutes prior to the exam. The steps for adding an exam are:

1. Click on the ***Exams*** tab left of the page and all the courses will appear in a tabular form.
2. Click on one of the ***Courses*** and an exams tab will appear.
3. Click on ***New Exam*** link to create a new exam.
4. Fill the exam date, time, duration with marks or a grade and click submit.
5. A rich text editor will appear with an optional attachment that can be added as shown in Figure 5.8.

6. Click on submit to create a new exam.

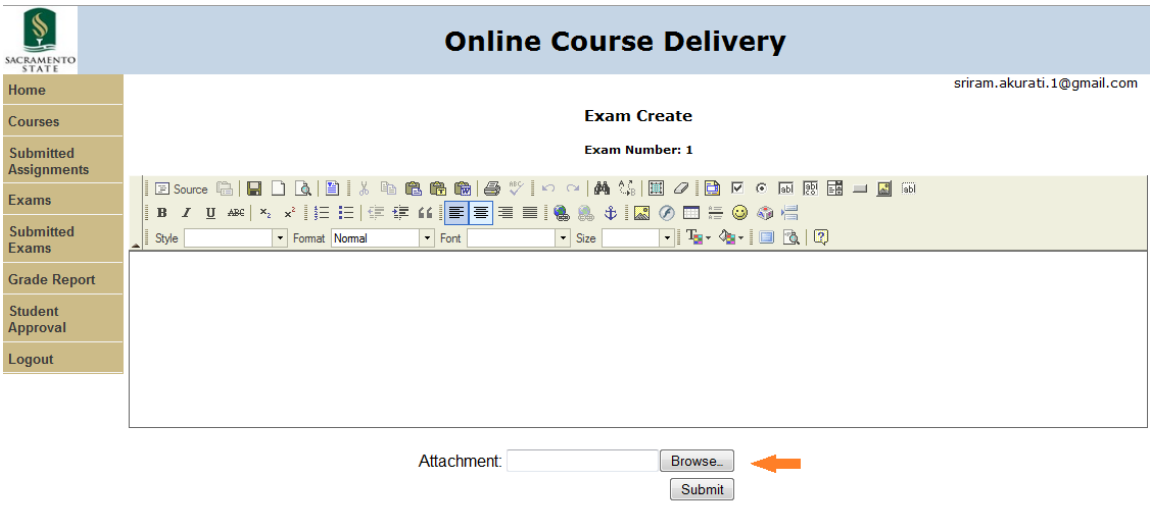

Figure 5.7 Create New Exam

5.1.9 Edit Exams and View Submitted Exams

Just as assignments, exams can also be edited. All the fields including time and date are editable. The exam content and the attachments can also be changed.

5.1.10 View Submitted Exams

Submitted exams by students can also be viewed and graded just as assignments.

5.1.11 Grade Report

A grade report is automatically generated for each student when he is approved by an instructor for a course. The grade report contains all the assignments, exams and the final

grade. Each individual's marks can be updated by clicking on the grade and filling the form that appears on the page as shown in Figure 5.9.

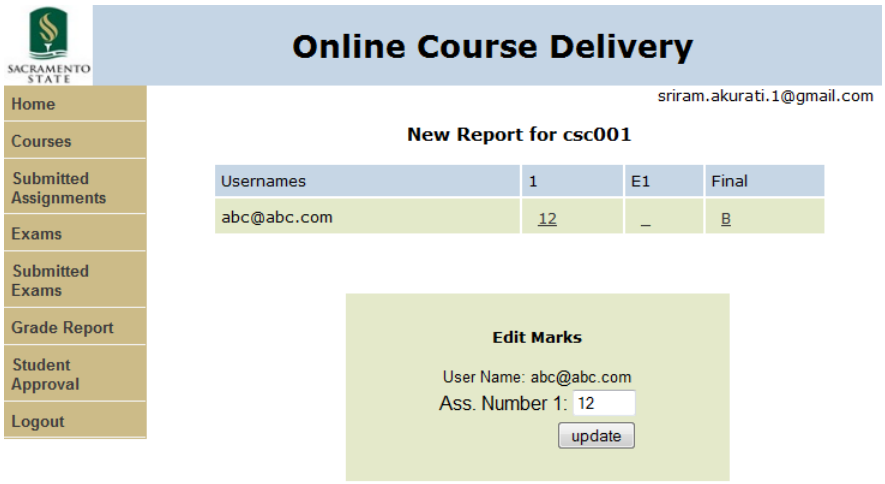

Figure 5.8 Grade Report

5.2 Student

The student is the end user in this web application. He can perform the following tasks:

1. Create an account without any approval from the administrator or the instructor.
2. Register for a course but only before the registration date and wait for an approval from the instructor.
3. View instructor posted assignment and submit assignments with attachments.
4. View lecture slides for each course.
5. Take time bound exams.
6. View grade reports for each course.

5.2.1 Create an Account

To create a student account a user can do the following:

1. Click on the ***Register*** link in the home page and fill the form with user type as student.
2. An activation link will be sent to the student's email id.
3. The user can proceed and click on the activation link which directs to a page where a new password can be entered.
4. Once the password is entered the user is registered and is ready to use the online course delivery application.

5.2.2 Register for Course

A student can register for course once he creates an account. The following sequence will show how to register for a course:

1. Click on ***Register*** link on the left of the page once logged in.
2. Click on the ***Course Number*** link which appears in the tabular column.
3. Once a tabular column appears with the details of the course click on the Register button as shown in Figure 5.10
4. Instructor approval is required for proceeding further.

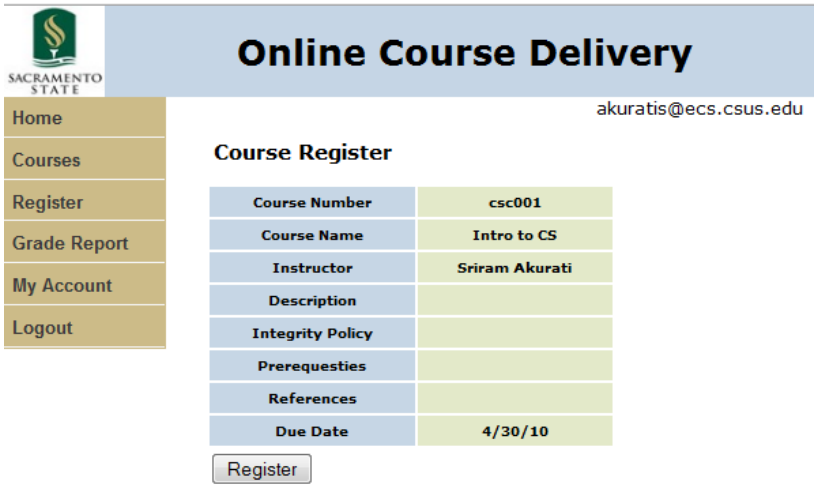

Figure 5.9 Student Register Course

5.2.3 Submit Assignments

Students can submit or view assignments created by the instructor. Each assignment consists of a due date and cutoff date. A student can submit an assignment even after the due date if he has not submitted it earlier. Once the cutoff date is crossed a student will no longer be able to submit assignments.

The procedure for submitting assignments is:

1. Click on the ***Courses*** link on the left of the page and select the ***Course Number.***
2. Once the assignments tab appears click on the ***assignment*** link which needs to be submitted.
3. Two rich text editors will be displayed one with the instructor assignment content and the other a blank one where the student can enter content as shown in Figure 5.11.

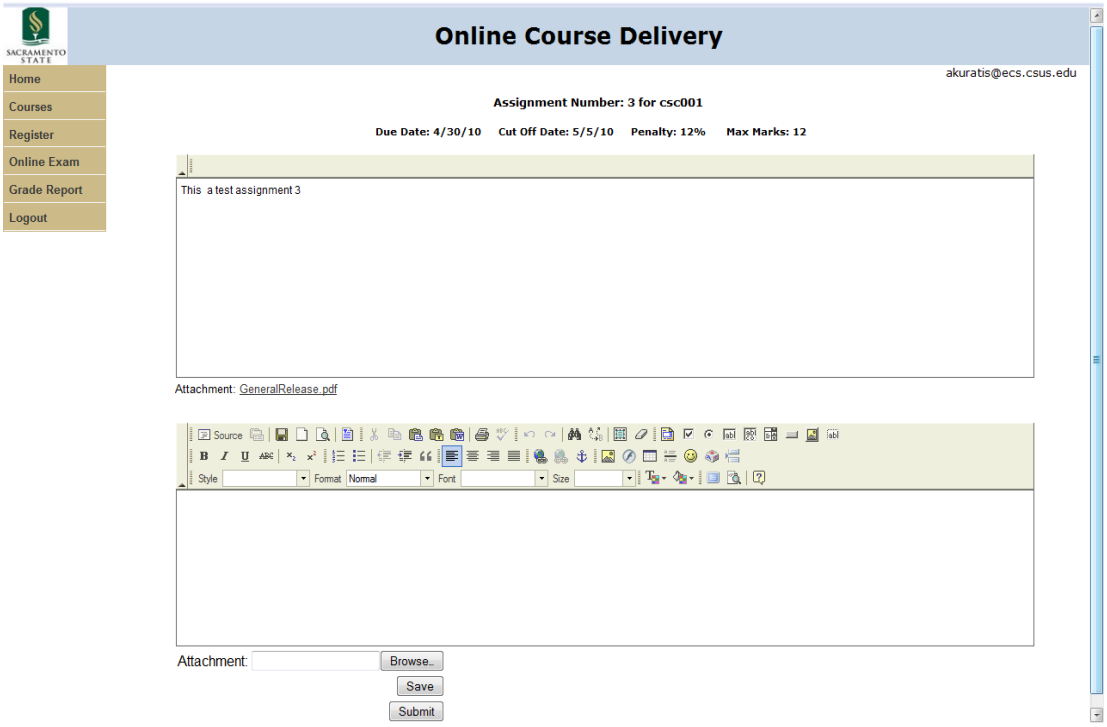

Figure 5.10 Student Submit Assignments

5.2.4 Take Online Exam

Once an exam is posted by an instructor it can be taken by a student registered for that particular course. The exam link opens 10 minutes prior to the scheduled time. A timer starts a countdown of the duration. Once countdown is complete the exam is submitted automatically. If the student checks in late the duration of the time is also decremented. Figure 5.12 shows the exam page with the countdown timer.

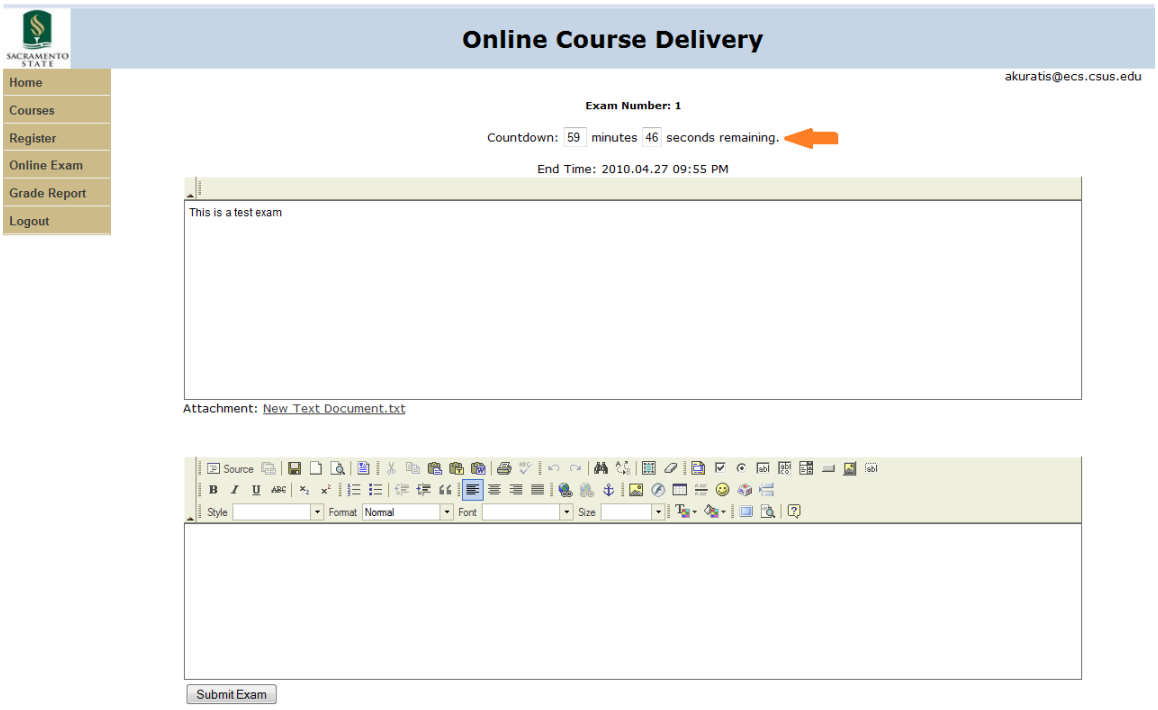

Figure 5.11 Student Submit Exam

5.2.4 Student Grade Report

The student can view his/her grade by clicking on the ***Grade Report*** link on the application. Each course has a table with all the assignments, exams and final grade posted. The grade report for a student is shown in Figure 5.13.

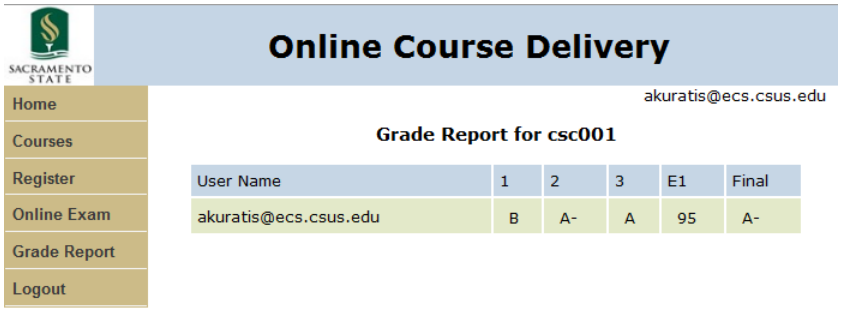

Figure 5.12 Student Grade Report

1. Click the ***Courses*** tab.
2. Press the **New** command button.
3. This opens the template for describing the course. The Figure 5.2 shows the template for creating any course.

5.3 Administrator

In this online course delivery project the administrator has the authority over creation of instructor accounts and deletion of both instructor and student accounts. The administrator can perform the following actions:

1. Check instructor's credentials and approve instructor accounts.
2. Delete instructor's and student's accounts.

5.3.1 Approve Instructor's Account

The administrator can approve instructor's account by clicking on ***Instructors Account*** link on the left of the page. A tabular form appears with the current approved instructors and requested account instructors. Then click on the instructor whose account needs to be approved. A new page opens with details of the instructor with an approve and deny button. Clicking on approve will send a mail to the instructor informing him that the account is approved with an activation link. Clicking on deny will also send a mail to the instructor about his account being denied. The details of the account will also be deleted when account is denied by the instructor.

5.3.2 Delete Instructor's and Student's Accounts

The administrator can also delete accounts of students and instructors. He can click on the students account or instructors account on his page. A tabular form appears with the student usernames or instructor usernames. Clicking on the username will lead to a form where the information of student or instructor is present with a delete button. Clicking on delete will delete all the student information and instructors' information including assignment, exams and lecture slides.

Chapter 6

FUTURE WORK AND SUMMARY

The online course delivery web application has most of the features necessary for conducting a course online. However, this application can be enhanced to bring in features like:

- Automated testing management
- Automated Grading system
- Real time chat
- Discussion Forums

Automated testing management: Automated testing feature can be included in a future project which can automatically grade assignment or exams based on a predefined set of questions and answers.

Automated Grading system: This feature is kind of extension to automated testing management. With this feature, the application grades the exam and assigns the marks to the Student. This feature will be more relevant for multiple-choice examinations.

Real time chat: A real time chat module with both voice and video can be enabled in a future application. This feature will greatly enhance this application by providing a face to face interaction with the students and the instructor. If the application can also store the chats, it will add a new dimension to the project.

Discussion Forums: Discussion Forums enable students to interact with each other.

To summarize, this application can be used by educational system. With this application, Instructor can create course, post assignments, exams and also answer student queries posted in discussion forums. The Student can use the system to take courses online, which includes taking assignments, exams and viewing lecture slides.

BIBLIOGRAPHY

[1] Brown Dan, Davis Michael Chad, Stanlick Scott "Struts 2 in Action", Manning Publications, 2008.

[2] Introduction to Spring Framework Development by Rod Johnson [Online]. Available: www.theserverside.com/tt/articles/article.tss?l=SpringFramework

[3] MySQL Community Server, [Online]. Available: "http://www.mysql.com/downloads/mysql/"

[4] "Online course delivery system using Salesform.com platform " Sirisha Mallempalli, Master's Project, Spring 2008,California State University, Sacramento.

[5] Sacramento State College of Continuing Education: Online Education http://www.cce.csus.edu/programs/online/

[6] Walls Craig, Breidenbach Ryan "Spring in Action" 2nd Edition, Manning Publications, 2008.